IEEE INFORMATION MODEL STANDARDS FOR TEST AND DIAGNOSIS

John Sheppard
ARINC
2551 Riva Road
Annapolis, MD 21401
410-266-2099
jsheppar@arinc.com

Mark Kaufman
NWAS
PO Box 5000
Corona, CA 91718
909-273-5725
kaufman.mark@corona.navy.mil

Abstract *- In this paper we discuss the use of information modeling to develop information exchange standards and metrics for test and diagnostics. For example, fault information is being transferred from one diagnostic reasoner to another. How many attributes does a fault have? One, three, fifteen? How do these attributes relate to the fault, to diagnosis, and tests? If both reasoners do not understand the "model" of a fault, then there will not be an information exchange. Humans can, when they are aware of the differences, resolve the mismatch in information. Computer programs cannot "talk" until they resolve small differences in conceptual information. We present an overview of the current and future directions of the Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) standards. We present a summary of the work completed so far on the diagnostic reasoner exchange standard and on the testability metrics standard.*

We address objectives, document organization, information modeling, service versus API specification, and other issues raised by the AI-ESTATE community. We also discuss the vision of the AI-ESTATE subcommittee in its work to integrate the AI-ESTATE information models and projects such as testability/diagnosability assessment and test/maintenance feedback.

INTRODUCTION

The Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) standards are information exchange standards for test and diagnosis. The original standards, the 1232 series, developed a means of exchange of information between diagnostic reasoners. As the information models for the 1232 standards were developed, it became apparent that these models could be used for standardizing testability and diagnosability metrics. This paper will be discussing the test and diagnosis information exchange standards followed by the testability and diagnosability standards. Other applications of these models will be briefly described.

In 1998, the third of a series of three standards was published by the IEEE addressing issues in system-level diagnostics. IEEE Std 1232-1995 defines the architecture of an AI-ESTATE-conformant system and has been published as a "full-use" standard; however, this standard was published before the vision of AI-ESTATE was fully developed. IEEE Std 1232.1-1997 defines a knowledge and data exchange standard and was published as a "trial-use" standard. Trial-use indicates that it is preliminary in nature, and the standards committee is seeking comments from organizations attempting to implement or use the standard. In 1998, IEEE Std 1232.2-1998 was approved. Its publication, also as a "trial-use" standard is imminent. This standard formally defines a set of standard software services to be provided by a diagnostic reasoner in an open-architecture test environment. Since it is also a trial-use standard, comment and feedback are necessary here as well. The standards were developed using information modeling, resulting in the definition of four information models addressing static and dynamic aspects of the diagnostic domain. Further, the IEEE 1232 AI-ESTATE series of standards provide the foundation for precise and unambiguous testability and diagnosability metrics.

As systems became more complex, costly, and difficult to diagnose and repair, initiatives were started to address these problems. The objective of one of these initiatives, testability, was to make systems easier to test. Early on, this focused on having enough test points in the right places. As systems evolved, it was recognized that the system design had to include characteristics to make the system easier to test. This was the start of considering testability as a design characteristic. As defined in MIL-STD-2165, testability is "a *design characteristic* which allows the status (operable, inoperable, or degraded) of an item to be determined and the isolation of faults within the item to be performed in a timely manner." [1]. The purpose of MIL-STD-2165 was to provide uniform procedures and methods to control planning, implementation, and verification of testability during the system acquisition process by the Department of Defense (DoD). It was to be applied during all phases of system development—from concept to production to fielding. This standard, though deficient in some areas, provided useful guidance to government suppliers. Further, lacking any equivalent industry standard, many commercial system developers have used it to guide their activities although it was not imposed as a requirement.

In this paper, we present an overview of the current and future directions of the AI-ESTATE standards. We address objectives, document organization, information modeling, service versus Applications Program Interface (API) specification, and other issues raised by the AI-ESTATE community. We also discuss the vision of the AI-ESTATE subcommittee in its work to integrate the AI-ESTATE information models and projects such as testability/diagnosability assessment and test/ maintenance feedback.

A VISION FOR TEST AND DIAGNOSIS STANDARDS

Diagnosis

The vision of AI-ESTATE is to provide an integrated, formal view of diagnostic information as it exists in diagnostic knowledge bases and as it is used (or generated) in diagnostic systems. We assert that the whole purpose of testing is to perform diagnosis [2]. In justifying this assumption, we rely on a very general definition of diagnosis, derived from its Greek components (δια γιγνωσκω) meaning, "to discern apart." Given such a broad definition, all testing is done to provide information about the object being tested and to differentiate some state of that object from a set of possible states.

In support of this vision, the AI-ESTATE committee has been working on combining the existing standards into a single, cohesive standard. This "unified" standard provides formal specifications of all of the information models (both for file exchange and for diagnostic processing), from which the service specifications are then derived and specified. The architectural framework is retained at the conceptual level to emphasize that a wide variety of implementation models are possible that still support standard exchange of information as long as the definition of that information is clear and unambiguous. Thus, in a sense, the models define the architecture, and the implementation is left entirely to the implementer.

With this vision in mind, we believe AI-ESTATE plays a central role in any test environment (thus the "All Test Environments" part of the name). To date, the focus of the standards has been the development of specifications supporting diagnosis in the traditional sense of the word (i.e., fault isolation). However, the broader context within which AI-ESTATE is envisioned to participate involves tying diagnostic information to explicit product behavior descriptions, assessments of the ability of testing to satisfy its requirements, and maturation of the diagnostic process through test and maintenance information feedback.

Testability

In 1997, the AI-ESTATE committee began to work on a new standard focusing on replacing the cancelled MIL-STD 2165. The military standard focused on specifying the essential elements of a testability program and explained the elements needed to define a testability program plan. In addition, MIL-STD 2165 included the "definition" of several testability metrics, including a testability checklist to be used to determine overall system testability. With the cancellation of military standards and specifications by the Perry Memo in 1994 [3], and with the lack of specificity and clarity in MIL-STD 2165, it became evident that a replacement was necessary.

The approach being taken to develop this standard involves defining testability and diagnosability metrics based on standard information models. Specifically, it was found that the AI-ESTATE models provided an excellent foundation for defining these metrics

THE AI-ESTATE ARCHITECTURE

According to IEEE Std 1232-1995, the AI-ESTATE architecture is "a conceptual model" in which "AI-ESTATE applications may use any combination of components and intercomponent communication" [4]. On the other hand, according to IEEE Std 1232.2-1998, AI-ESTATE includes explicit definitions of services to be provided by a diagnostic reasoner, where the services "can be thought of as responses to client requests from the other components of the system architecture" [5]. More specifically, "each of the elements that interface with the reasoner will interact through [an] application executive and will provide its own set of encapsulated services to its respective clients" [5].

Although not necessarily obvious from the standards themselves, these two "views" of the AI-ESTATE architecture present an interesting dichotomy. Specifically, the architecture standard provides a concept of AI-ESTATE that permits any communication mechanism to be used between components of a test environment in support of the diagnostics provided by that environment. The service specification, on the other hand, seems to cast the communication mechanism in the form of a client-server architecture.

We note that the intent of AI-ESTATE is to provide a formal, standard framework for the exchange of diagnostic information (both static and dynamic) in a test environment. This exchange occurs at two levels. At the first level, data and knowledge are exchanged through a neutral exchange format, as specified by IEEE Std 1232.1-1997 [6]. At the second level, specified by IEEE Std 1232.2-1998 [5]information is exchanged as needed between software applications within the test environment. This information includes entities from a model or information on the current state of the diagnostic process.

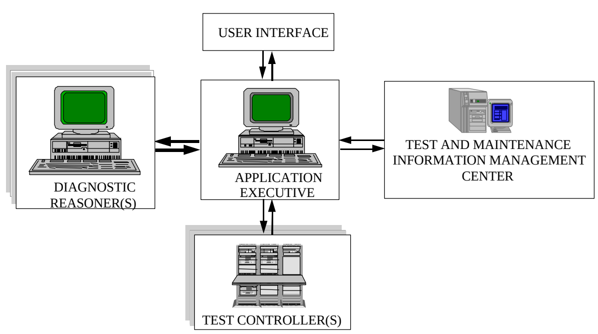

Figure 1. AI-ESTATE Architecture

To facilitate encapsulation of the information and the underlying mechanisms providing that encapsulation, AI-ESTATE assumes the presence of an "application executive." We emphasize that this application executive need not be a physically separate software process but can be identified as a "view" of the software process when it involves the communication activity. This view of the architecture is shown in Figure 1. In the following sections, we will provide a more detailed discussion of the exchange and service elements of the architecture.

Data and Knowledge Exchange

ISO 10303–11 (EXPRESS) and ISO 10303–12 (EXPRESS-I) are used to define information models and exchange formats for diagnostic knowledge [7], [8]. These international standards are being maintained by the STEP (Standard for the Exchange of Product model data) community. The current approach to static information exchange within AI-ESTATE is to derive the exchange format from the formal information models as specified in the ISO standards.

The purpose of information modeling is to provide a formal specification of the *semantics* of information that is being used in an "information system." Specifically, information models identify the key entities of information to be used, their relationships to one another, and the "behavior" of these entities in terms of constraints on valid values [9]. The intent is to ensure that definitions of these entities are unambiguous.

For example, central to the test and diagnosis problem is the definition of a “test.” If we ask a digital test engineer what a test is, it is possible that the answer will be something like “a set of vectors used to determine whether or not a digital circuit is working properly.” On the other hand, if we ask a diagnostic modeler what a test is, the answer is likely to be “any combination of stimulus, response, and a basis for comparison that can be used to detect a fault.”

On the surface, these two definitions appear very similar; however, there is a fundamental difference. For the digital test engineer, there is an implicit assumption that a “test” corresponds to the entire suite of vectors. For the diagnostic modeler, individual vectors are tests as well.

As a similar example, the test engineer and diagnostic modeler are likely to have different definitions for “diagnosis.” The act of doing diagnosis, for most test engineers, corresponds to running tests after dropping off of the “go-path.” For the diagnostic modeler, since “no fault” is a diagnosis, the entire test process (including the go-path) is part of doing diagnosis.

It may appear that we are “splitting hairs,” but formal definition of terms and information entities is an exercise in splitting hairs. Further, such hair-splitting is essential to ensure that communication is unambiguous—especially when we are concerned with communication between software processes. No assumption can go unstated;

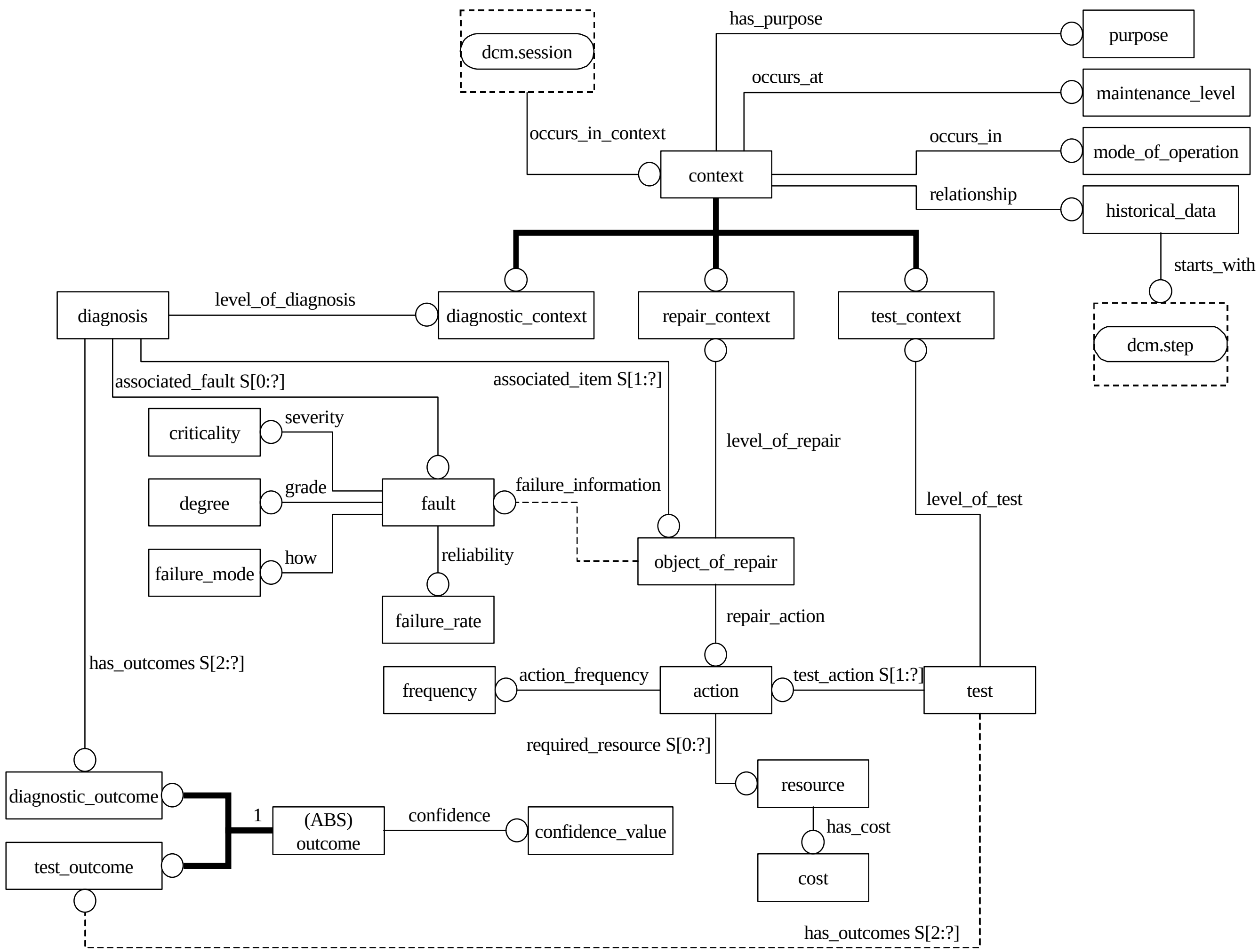

Figure 2. Revised Common Element Model

otherwise, the risk exists that something will be misunderstood. Information models formally state all of the assumptions.

A New Information Model

When IEEE 1232.1 was published, it was published as a "trial-use" standard to provide a period for people to study it, attempt to implement it, and provide feedback to the AI-ESTATE committee on the ability of the standard to satisfy the stated requirements. Since publication, comments have been received to indicate that ambiguity still exists in the information models.

Because of the concern that the information models are still ambiguous, the models are undergoing close examination and modification. It is interesting to note that much of the ambiguity has been identified in connection with a related standard being developed by the AI-ESTATE committee—P1522 Standard for Testability and Diagnosability Metrics and Characteristics. AI-ESTATE's approach to developing this new standard involved defining the metrics based on the information models within the P1232 standard. As we were identifying metrics to be standardized, we discovered that the current models were incapable of supporting their definition.

A conceptual view of the revised common element model is shown in Figure 2. Of note in the revised model is the addition of a context entity and the differentiation between fault and function. Many diagnostic tools are highly context dependent (e.g., different procedures are suggested based on the environmental conditions of the test or the skill levels of the maintenance technicians). In addition, several tools focus on modeling function rather than physical faults to support modeling at the system level. Since the distinctions among context and type of analysis were not previously made explicit, new entities were defined to eliminate ambiguity that may arise from different approaches and contexts for modeling.

Diagnostic Services

The approach taken to defining services in AI-ESTATE has been based on the traversal (i.e., the following of the relationships defined between model entities to access specific pieces of information in the models) of the information models. The "simplest" services involve traversing the models defined in IEEE 1232.1 (i.e., the exchange models); however, these models provide little functionality in terms of actual diagnosis.

In IEEE 1232.2, a novel use of information modeling was applied in that a dynamic information model was specified to support dynamic services. This model, called the "dynamic context model," relied on dynamically creating entities that populate the model during a diagnostic session. In fact, as suggested by "dcm.session" and "dcm.step" in the model shown in Figure 2, a diagnostic session is modeled as a sequence of steps instantiated from the set of possible values specified in the static model. Details of how the service specification is expected to be implemented can be found in [10], [11].

One of the concerns raised by a member of the AI-ESTATE committee was whether the standard specifies a set of services or simply an Application Programming Interface. The claim was that the service specification must include a behavior specification as well and that this can only be accomplished by defining a set of baseline behaviors, perhaps through some sort of test bed.

The committee observed that people have different opinions over the difference between a service specification and an API specification. Many, in fact, took issue with the claim that they were different. Further, it was determined that including test cases to specify standard behavior was not desirable in this context due to the wide variety of diagnostic approaches using common diagnostic knowledge. Rather, it was believed that it was more important for the information itself to be standardized and the specific behavior to be left to the implementation.

Testability and Diagnosability Metrics

Testability has been broadly recognized as the "-ility" that deals with those aspects of a system that allow the status (operable, inoperable, or degraded) or health state to be determined. Early work in the field primarily dealt with the design aspects such as controllability and observability. Almost from the start this was applied to the manufacturing of systems where test was seen as a device to improve production yields. This has been slowly expanded to include the aspects of field maintainability such as false alarms, isolation percentages, and other factors associated with the burden of maintaining a system.

In the industry, many terms such as test coverage and Fraction of Fault Detection (FFD) are not precisely defined or have multiple definitions. Further, each diagnostic tool calculates these terms differently and therefore the results are not directly comparable. Some measures, such as false alarm rate, are not measurable in field applications. Other measures such as Incremental Fault Resolution, Operational Isolation, and Fault Isolation Resolution appear different, but mean nearly the same thing.

Lacking well-defined testability measures, the tasks of establishing testability requirements, and predicting and evaluating the testability of the design are extremely difficult. This in turn makes effective participation in the design for testability process difficult. These difficulties will be greatly diminished by the establishment of standard testability metrics. An immediate benefit will come with a consistent, precise, and measurable set of testability attributes that can be compared across systems, tools, and within iterations of a system's design.

MIL-STD-2165 did not have precise and unambiguous definitions of measurable testability figures-of-merit and relied mostly on a weighting scheme for testability assessment. (It should be noted, however, that the standard did permit the use of analytical tools for testability assessment such as SCOAP, STAMP, and WSTA).

As we strive to establish concurrent engineering practices, the interchange between the testability function and other functions becomes even more important. To create integrated diagnostic environments, where the elements of automatic testing, manual testing, training, maintenance aids, and technical information work in concert with the testability element, we must maximize the reuse of data, information, knowledge, and software. Complete diagnostic systems include Built-In-Test (BIT), Automatic Test Equipment (ATE), and manual troubleshooting. It would be desirable to be able to predict and evaluate the testability of systems at these levels.

It is not an accident that the P1522 standard contains both the word testability and the word diagnosability. The distinction is not always easy to maintain, especially in light of the expansion of the use of the testability term. Figure 3 shows the basic relationship, with diagnosability being the larger term and encompassing all aspects of detection, fault localization, and fault identification. The boundary is fuzzy and often it is not clear when one term applies and the other does not. The P1522 standard is meant to encompass both aspects of the test problem. Because of the long history of the use of the testability term, we will seldom draw a distinction. However, the use of both terms is significant in that testability is not independent of the diagnostic process. The writing of test procedures cannot and should not be done separately from testability analyses. To do so, would be meeting the letter of the requirements without considering the intent.

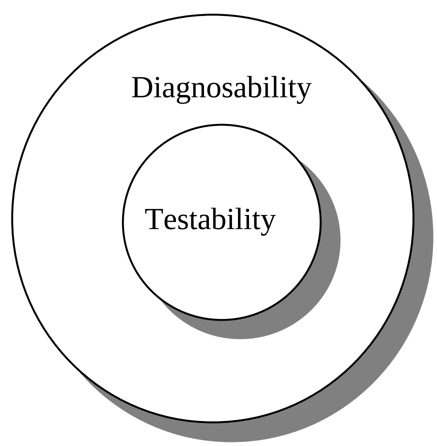

Figure 3. Relationship Between Diagnosability and Testability

TESTABILITY AND DIAGNOSABILITY METRICS OBJECTIVES

It is the objective of the P1522 standard to provide notionally correct, inherently useful, and mathematically precise definitions of testability metrics and characteristics. It is expected that the metrics may be used to either measure the testability of a system, or predict the testability of a system. Notionally correct means that the measures are not in conflict with intuitive and historical representations. Beyond that, the measures must be either measurable or predictable. The former may be used in the specification and enforcement of acquisition clauses concerning factory and field-testing, and maintainability of complex systems. The latter may be used in an iterative fashion to improve the factory and field-testing and maintainability of complex systems. The most useful of all are measures that can be used for both. Because of the last point, the emphasis will be on measurable quantities (metrics).

Things that can be enumerated by observation and folded into the defined figures-of-merit will be developed into metrics. However, a few measures are inherently useful on the design side even if they are not measurable in the field and they are defined in a separate section in P1522. The end purpose is to provide an unambiguous source for definitions of common and uncommon testability and diagnosability terms such that each individual encountering the metric can know precisely what that metric measures.

ISSUES

MIL-STD-2165 defined Fraction of Faults Detected (FFD) two ways. The first is the fraction of *all* faults detected by BIT/External Test Equipment (ETE). The second is the fraction of *all detectable* faults detected by BIT/ETE [1]. False alarms were excluded from the definition. From these two variations grew many others. As noted in “Organizational-Level Testability” [12] FFD can be defined as:

- Fraction of all faults detected or detectable by BIT/ETE
- Fraction of all detectable faults detected or detectable with BIT/ETE
- Fraction of all faults detected through the use of defined means. Defined means implies all means of detection that have been identified.
- Percentage of all faults automatically detected by BIT/ETE
- Percentage of all faults detectable by BIT/ETE
- Percentage of all faults detectable on-line by BIT/ETE
- Percentage of all faults and out-of-tolerance conditions detectable by BIT/ETE
- Percentage of all faults detectable by any means

One problem with traditional metrics is that they are “overloaded.” Overloaded in this case means that due to “common understanding” of the terms, fine variations are not specified. Consequently, users of the term do not necessarily know the implications of a precise definition. Discussions of overloaded terms go on at length, in part because everyone in the discussion has brought along a lot of mental baggage. Often, progress is only made when a neutral term is chosen and the meaning is built from the ground up. This overloading is so severe, for example, that there was no definition of FFD in *System Test and Diagnosis* [2], the authors preferring to use Non-Detection Percentage (NDP). FFD is the negative of NDP and is equal to 1–NDP.

Even the number of faults counted in the field require a more precise definition. The “overloaded” version is simply a count of all the things that failed. The quantity of all faults, as usually defined in the industry, is different. The quantity of faults detected by BIT/ETE, and the quantity of faults detected exclude the occurrence of false alarms. Intermittent faults are classified as a single fault. Temporary faults, those caused by external transients of noise, are not classified as faults.

```
FUNCTION ffd(model:EDIM.edim; lvl:CEM.level) : REAL;
      LOCAL
            diag_count : INTEGER;
            diags : SET [0:?] OF EDIM.inference
            detect_set : SET [0:?] OF CEM.diagnosis := NULL;
      END_LOCAL;

      diag_count := SIZEOF(QUERY(tmp <* model.model_diagnosis |
      tmp.level_of_diagnosis = lvl);
      REPEAT I := LOINDEX(model.inference) TO HIINDEX(model.inference);
            diags := QUERY(tmp <* model.inference[I].conjuncts |
                        (TYPEOF(tmp) = 'EDIM.diagnostic_inference'));
            diags := diags + QUERY(tmp <* model.inference[i].disjuncts |
            IEEE Std 1232.1-1997diags := QUERY(tmp <* diags |
                        tmp.pos_neg = negative OR
                        NOT(tmp.diagnostic_assertion = 'Good'));
            detect_set := detect_set + QUERY(tmp <* diags.for_diagnosis |
                        tmp.level_of_diagnosis = lvl);
      END_REPEAT;
      RETURN(SIZEOF(detect_set) / diag_count);


END_FUNCTION;
```

Figure 4. Sample Metric Definition in EXPRESS

Another aspect of the challenge is that many metrics sound different but are not. Below are some examples.

- *Ambiguity Group Isolation Probabilities* is the cumulative probability that any detected fault can be isolated by BIT or ETE to an ambiguity group of size L or less.
- *Fault Isolation Resolution* is the cumulative probability that any detected fault can be isolated to an ambiguity group of a targeted size or less.
- *Isolation Level* is the ratio of the number of ambiguity groups to the total number of isolatable components.

- *System Operational Isolation Level* is the percentage of observed faults that result in isolation to *n* or fewer replaceable units.

All of these terms were and are valuable. The value of these terms will be increased with precise meanings for each one.

ASSUMPTIONS

The development of a diagnostic capability includes system level analysis. As such, it is assumed that a system level approach is undertaken, and those diagnostic strategies and testability criteria have been explicitly developed or at least structured. These may be variables in the formulation, but cannot be completely undefined. The primary assumptions are twofold and deal with inherent usefulness from prior experience and the ability to precisely define the term from first principles. In some cases, we will assume the existence of a diagnostic model such as one based on the IEEE 1232 series of standards. Metrics will be derived from the entities and attributes based on these information models. In other cases, we will rely on a demonstrated ability to measure items related to the testing at the design, factory, and field levels concerning the maintainability of complex systems. In the latter case, information models will be developed as necessary to define all relevant entities and attributes.

Each term carries with it a number of additional assumptions (such as single or multiple failure) and is explicitly dealt with on a term by term basis in the section on metrics and characteristics.

The FFD metric assumes the existence of a diagnostic model that ties tests (especially test outcomes) to potential faults in the system analyzed. Within AI-ESTATE, tests, diagnoses, and faults are modeled explicitly in the common element model. In addition, AI-ESTATE includes specifications for two diagnostic models—the fault tree model and the Enhanced Diagnostic Inference Model (EDIM). Due to its generality, the EDIM was used to define FFD.

The assumptions used to define FFD are as follows:

- We are interested in the various metrics at a particular level;
- A hierarchical element exists at a particular level;
- No descendant of a hierarchical element is at the same level as that hierarchical element; and
- At this point, we don't care about the ordering of the levels.

As an example, one metric defined using the model is Fractions of Faults Detected (FFD).

From these assumptions and the information models, we can define FFD using the procedural constructs of EXPRESS. Specifically, a function (FFD) can be specified as in Figure 4. In the process of defining this one metric, several issues were identified. These issues are discussed in detail in [13].

OTHER APPLICATIONS

Ties to Maintenance Feedback

In 1993, a Project Authorization Request (PAR) was submitted to the IEEE for new standards project related to specifying information and services for test and maintenance information feedback. The Test and Maintenance Information Management Standard (TMIMS) project was approved by the IEEE in early 1994. The focus of this project was to define exchange and service standards (similar to AI-ESTATE) which support the test and diagnostic maturation process. In 1998, due to a lack of progress, the TMIMS PAR was cancelled.

AI-ESTATE continues to require definition of exchange and service standards related to test and maintenance information. In 1998, shortly after the cancellation of the TMIMS PAR, the AI-ESTATE committee decided to include test and maintenance information in its scope. The approach will be consistent with AI-ESTATE (i.e., the definition of information models and EXPRESS-level services derived from traversing the models). Further, it is anticipated that the starting point for the new models will be the dynamic context model in IEEE 1232.2. By keeping track of the sequence of events during a diagnostic session, much of the historical information is identified and captured that can be used for later diagnostic maturation.

Ties to Product Descriptions

Through the 1990s, the IEEE has been developing a family of standards under the umbrella of “A Broad Based Environment for Test” (ABBET) [14], [15]. An early architecture of ABBET, based on information modeling, presented ABBET as five layers: 1) product description, 2) test requirements/strategy, 3) test methods, 4) test resources, and 5) instrumentation. Since then, standards for the “lower layers” of ABBET (i.e., layers 3–5) have been defined; however, it has long been recognized that the major benefit from standardization will come from the “upper layers” (i.e., layers 1 and 2).

AI-ESTATE satisfies many of the requirements related to layer two of ABBET (however, AI-ESTATE has never been considered part of the ABBET family). Further, a recent proposal for a new information model-based standard, called the Test Requirements Model (TeRM), will address specific concerns of test requirements [16], [17]. Standards for the product description layer have always been problematic due to issues related to the revelation of intellectual property. With the combination of TeRM, AI-ESTATE, and TMIMS, it is anticipated that intellectual property can be hidden from information provided in standard form while still supporting the test engineer fully.

CONCLUSION

Reasoning system technology has progressed to the point where electronic and other complex systems are employing artificial intelligence as a primary component in meeting system test and verification requirements. This is giving rise to a proliferation of AI-based design, test, and diagnostic tools. Unfortunately, the lack of standard interfaces between these reasoning systems has increased the likelihood of significantly higher product life-cycle cost. Such costs arise from redundant engineering efforts during design and test phases, sizeable investment in special-purpose tools, and loss of system configuration control.

The AI-ESTATE standards promise to facilitate ease in production testing and long-term support of systems, as well as reducing overall product life-cycle cost. This will be accomplished by facilitating portability, knowledge reuse, and sharing of test and diagnostic information among embedded, automatic, and stand-alone test systems within the broader scope of product design, manufacture, and support.

AI-ESTATE was first conceived in 1988 as a standard for representing expert-system rule bases in the context of maintenance data collection. Since that time, AI-ESTATE has evolved to be embodied in three published standards related to the exchange of diagnostic information and the interaction of diagnostic reasoners within a test environment. The three standards have been recommended for inclusion on the US DoD ATS Executive Agent’s list of standard satisfying requirements for ATS critical interfaces. In looking to the next generation, AI-ESTATE is expanding to address issues of testability, diagnosability, maintenance data collection, and test requirements specification.

ACKNOWLEDGMENTS

In many ways, it is unfortunate that a paper such as this includes only the names of two authors. The work reported here is the result of efforts of a committee of devoted volunteers who have supplied their expertise in system test and diagnosis to develop strong, sound standards supporting the diagnostics community. We would like to thank Les Orlidge, Randy Simpson, Tim Bearse, Tim Wilmering, Greg Bowman, Dave Kleinman, Lee

Shombert, Sharon Goodall, Len Haynes, Jack Taylor, and Helmut Scheibenzuber for their efforts in developing the standards and promoting their use in industry.

REFERENCES

[1] MIL STD 2165. 1985. *Testability Program for Electronic Systems and Equipment*, Washington, DC: Naval Electronic Systems Command (ELEX-8111

[2] Simpson, W. and Sheppard, J. 1994. *System Test and Diagnosis*, Boston, MA: Kluwer Academic Publishers.

[3] Perry, William. 1994. "Specifications and Standards—A New Way of Doing Business," US Department of Defense Policy Memorandum.

[4] IEEE Std 1232-1995. *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Overview and Architecture*, Piscataway, NJ: IEEE Standards Press.

[5] IEEE Std 1232.2-1998. *IEEE Trial-Use Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Service Specification*, Piscataway, NJ: IEEE Standards Press.

[6] IEEE Std 1232.1-1997. *IEEE Trial-Use Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE): Data and Knowledge Specification*, Piscataway, NJ: IEEE Standards Press.

[7] ISO 10303-11:1994. *Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 11: Description Methods: The EXPRESS Language Reference Manual*, Geneva, Switzerland: International Organization for Standardization.

[8] ISO 10303-12:1997. *Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 12: Description Methods: The EXPRESS-I Language Reference Manual*, Geneva, Switzerland: International Organization for Standardization.

[9] Schenk, D. and Wilson, P. 1994. *Information Modeling: The EXPRESS Way*, New York: Oxford University Press.

[10] Sheppard, J. and Maguire, R. 1996. "Application Scenarios for AI-ESTATE Services," *AUTOTESTCON '96 Conference Record*, New York: IEEE Press.

[11] Sheppard, J. and Orlidge, L. 1997. Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)—A New Standard for System Diagnostics," *Proceedings of the International Test Conference*, Los Alamitos, CA: IEEE Computer Society Press.

[12] Simpson, W., Bailey, J., Barto, K. and Esker, E., 1985 "Organization-Level Testability Prediction", ARINC Research Corporation Report 1511-01-3623 Prepared for the Rome Air Development Center.

[13] Kaufman, M. and Sheppard, J. 1999. "IEEE P1522—A Formal Standard in Testability and Diagnosability Assessment, *AUTOTESTCON '99 Conference Record*, New York: IEEE Press.

[14] IEEE Std 1226-1993. *IEEE Trial-Use Standard for A Broad Based Environment for Test (ABBET): Overview and Architecture*, Piscataway, NJ: IEEE Standards Press.

[15] IEEE Std 1226.6-1996. *IEEE Guide to the Understanding of A Broad Based Environment for Test (ABBET)*, Piscataway, NJ: IEEE Standards Press.

[16] Shombert, L. 1998. *Test Requirements Model Language Reference Manual*, Draft 0.1, Technical Report CAE-1998-07-01, Vienna, VA: Intermetrics.

[17] Shombert, L. and Sheppard, J. 1998. "A Behavior Model for Next Generation Test Systems," *Journal of Electronic Testing: Theory and Applications*, Vol. 13, No. 3.